BULLETIN OF THE POLISH
ACADEMY OF SCIENCES
TECHNICAL SCIENCES
Vol. 33, No. 9–10, 1985

VARIA
COMPUTER SCIENCE

Decision Tables and Decision Algorithms

by

Zdzisław PAWLAK

Presented by Z. PAWLAK on April 25, 1985

Summary. We show in this note the application of the rough set approach to decision tables analysis yields a simple method of checking whether the decision table is deterministic or not, and we also demonstrate how such an approach can be used to decision tables and decision algorithms simplification.

1. Introduction. The main objective of this note is to show that basic problems of decision tables analysis (see [4]) can be casly formulated and solved within the framework of rough set theory (see [2, 3]).

First, we define formally the notion of a decision table and next some basic properties of decision tables are stated. Further we introduce a decision language in which the concept of a decision algorithm is defined. We show that the correctnes of a decision algorithm can be proved in the language thus introduced.

The proposed approach has been applied to an expert system design and implementation, which controls the cement kiln operations (see [1]).

2. Decision tables

2.1. Basic definitions. A decision table is a system

$$S = (U, C, D, V, f)$$

where: U – is a set of **states**, called the **universe**, C and D are sets of **conditions** and **decisions** attributes, respectively, $V = \bigcup_{a \in C \cup D} V_a$, and V_a – is the set of **values** of attribute a, or **domain** of an attribute a, $f: U \times C \cup D \to V$ is a **decision function** (total). We assume that sets U, C and D are not empty and V_a is at least two-element set for every attribute.

The function $f_x: C \cup D \to V$, such that $f_x(a) = f(x, a)$ for every $x \in U$ and $a \in C \cup D$ will be called the **decision rule** in S (the semantical decision rule).

If f_x is a decision rule in S then f_x/C and f_x/D are called **condition** and **decision** of f_x, respectively.

A decision rule f_x is **deterministic** in S if for every $y \neq x$ $f_x/\tilde{C} = f_y/\tilde{C}$ implies $f_x/D = f_y/D$; otherwise f_x is **nondeterministic**.

A decision table is **deterministic (consistent)** if all its decision rules are **deterministic**; otherwise a decision table is **nondeterministic (inconsistent)**.

Let $S = (U, C, D, V, f)$ be a decision table and let $B \subseteq C \cup D$. The B-**restriction** of S is a system $S/B = (U, E, F, W, g)$, where $B = E \cup F$, $W = \bigcup_{a \in B} V_a$, $g = f/U \times B$.

Let $S = (U, C, D, V, f)$ be a decision table and let $X \subseteq U$. The X-**restriction** of S is a decision table $S/X = (X, C, D, V', f')$, where

$$V' = \bigcup_{a \in C \cup D} V_{a,X}$$

$$V_{a,X} = \{v \in V_a: \text{there exists } x \in X \text{ such that } f_x(x, a) = v\}$$

$$f' = f/X \times C \cup D.$$

An example of a decision table No. 1 is shown below.

U	a	b	c	d	e
1	1	0	2	2	0
2	0	1	1	1	2
3	2	0	0	1	1
4	1	1	0	2	2
5	1	0	2	0	1
6	2	2	0	1	1
7	2	1	1	1	2
8	0	1	1	0	1

Decision Table No. 1

where a, b, c and d, e are conditions and decisions attributes, respectively.

2.2. Indiscernibility relation. With every subset of attributes $B \subseteq C \cup D$ in S we associate an **indiscernibility relation** $\tilde{B} \subseteq U \times U$, defined as follows:

$$(x, y) \in \tilde{B} \text{ iff } f_x(a) = f_y(a), \text{ for every } a \in B \text{ and } x, y \in U.$$

If $(x, y) \in \tilde{B}$ we say that x and y are B-**indiscernible** in S.

It is obvious that $\tilde{B}$ is an equivalence relation for any $B \subseteq C \cup D$.

Equivalence classes of the relation $\tilde{B}$ are called B-**elementary sets** in S.

The subset $X \subseteq U$ is B-**definable** in S if it is a finite union of B-elementary sets in S.

For example, in the decision table No. 1 states 2 and 8 are a-indiscernible and states 3 and 6 are $\{d, e\}$-indiscernible.

2.3. Approximations of sets. Let $S = (U, C, D, V, f)$ be a decision table and let $B \subseteq C \cup D$, $X \subseteq U$.

The B-**lower** and B-**upper aproximation** of X in S are defined as follows:

$$\underline{B}X = \{x \in U : [x]_{\tilde{B}} \subseteq X\}$$

$$\bar{B}X = \{x \in U : [x]_{\tilde{B}} \cap X \neq \emptyset\}.$$

The set

$$Bn_B(X) = \bar{B}X - \underline{B}X$$

will be called the B-**boundary** of X in S.

It is easy to see that set $X \subseteq U$ is B-definable in S iff $\bar{B}X = \underline{B}X$.

The B-**lower** and B-**upper approximation** of a family of subsets of U $\mathscr{X} = \{X_1, X_2, \ldots, X_k\}$, $X_i \subseteq U$ is defined as follows:

$$\underline{B}\mathscr{X} = \underline{B}X_1, \underline{B}X_2, \ldots, \underline{B}X_k\}$$

$$\bar{B}\mathscr{X} = \{\bar{B}X_1, \bar{B}X_2, \ldots, \bar{B}X_k\}.$$

If $B \subseteq C \cup D$, then B^* denotes the family of all equivalence classes of the relation $\tilde{B}$. Let $A, B \subseteq C \cup D$. The A-**positive region** of B^* in S is the set

$$\mathrm{Pos}_A(B^*) = \bigcup_{i=1}^{n} \underline{B}X_i$$

and the A-**doubtful region** of B^* in S is the set

$$Bn_A(B^*) = \bigcup_{i=1}^{n} Bn_A X_i$$

where $B^* = \{X_1, X_2, \ldots, X_n\}$.

Certainly, $\mathrm{Pos}_A(B^*) \cup Bn_A(B^*) = U$.

The number

$$\gamma_A(B^*) = \frac{\text{card Pos}_A(B^*)}{\text{card } U}$$

will be called a **quality** of the **approximation** of B^* by A in S, and the number

$$\beta_A(B^*) = \frac{\text{card Pos}_A(B^*)}{\sum_{i=1}^{n} \text{card } \bar{A}X_i}$$

will be called the **quality of the approximation** of B^* by A in S.

Obviously

$$0 \leqslant \beta_A(B^*) \leqslant \gamma_A(B^*) \leqslant 1.$$

It is obvious that the decision table S is deterministic iff $\gamma_C(D^*) = 1$ (or $\beta_C(D^*) = 1$).

2.4. Dependency of attributes. Let $S = (U, C, D, V, f)$, $A, B \subseteq C \cup D$ and $0 \leqslant k \leqslant 1$. We say that set of attributes B **depends in degree** k (k-depends) **on set** A in S, in symbols $A \xrightarrow{k} B$, if $k = \gamma_A(B^*)$.

Thus, the decision table $S = (U, C, D, V, f)$ is deterministic iff the set of decision attributes D 1-depends on the set of condition attributes C.

From the above there follows an important property:

Property 1. Each decision table $S = (U, C, D, V, f)$ can be decomposed into two decision tables $S/\mathrm{Pos}_C(D^*)$ and $S/Bn_C(D^*)$, such that $C \xrightarrow{1} D$ in $S/\mathrm{Pos}_C(D^*)$ and $C \xrightarrow{0} D$ in $S/Bn_C(D^*)$.

Thus, each decision table can be decomposed into two decision tables (possibly empty) such that one table is deterministic and the second is nondeterministic and does not contain the deterministic subtable.

For example, the decision table No. 1 is nondeterministic and can be decomposed into two tables:

U	a	b	c	d	e
3	2	0	0	1	1
4	1	1	0	2	2
6	2	2	0	1	1
7	2	1	1	1	2

Decision Table No 2

U	a	b	c	d	e
1	1	0	2	2	0
2	0	1	1	1	2
5	1	0	2	0	1
8	0	1	1	0	1

Decision Table No. 3

The decision table No. 2 is deterministic and the decision table No. 3 is nondeterministic.

2.5. Reduction of attributes. We say that a subset of attributes $B \subseteq C \cup D$ is **independent** in $S = (U, C, D, V, f)$ if for every $A \subset B$, $\tilde{A} \supseteq \tilde{B}$; otherwise the subset B is **dependent** in S.

That is, $B \subseteq C \cup D$ is dependent in S if there exists $A \subset B$ such that $\tilde{A} = \tilde{B}$.

Set $A \subseteq B \subseteq C \cup D$ is a reduct of B in S if A is the least independent set in B.

If the only reduct of B in S is B itself, we say that the decision table S is B-reduced.

If the decision table $S = (U, C, D, V, f)$ is C-reduced we say that S is reduced.

It is easy to see that the decision table No. 1 is not reduced, and the set of condition attributes is dependent. The set of control attributes C has only one reduct, which is the set $\{a, b\}$. Thus, the decision table No. 1 can be reduced to table No. 4.

U	a	b	d	e
1	1	0	2	0
2	0	1	1	2
3	2	0	1	1
4	1	1	2	2
5	1	0	0	1
6	2	2	1	1
7	2	1	1	2
8	0	1	0	1

Decision Table No 4

3. The decision language.

3.1. Syntax of the decision language. With each decision table $S = (U, C, D, V, f)$ we associate a **decision language** L_S.

The set of **terms** T_S in L_S is the least set satisfying the conditions:

A1) Constans 0.1 are terms in L_S

A2) Any expression of the form $(a := v)$, where $a \in C \cup D$, $v \in V_a$ is a term in L_S

A3) If t and s are terms in L_S, so are $-t$, $(t+s)$ and $(t \cdot s)$.

The set of **formulas** F_S in L_S is the least set satisfying the conditions:

B1) Constans T (true) and F (false) are formulas in L_S

B2) If t and s are terms in L_S, then $t = s$ and $t \Rightarrow s$ are formulas in L_S

B3) If Φ and Ψ are formulas in L_S, then $\sim \Phi$, $(\Phi \vee \Psi)$ and $(\Phi \wedge \Psi)$ are formulas in L_S.

For example,

$$-((a := 1)(b := 0) + (c := 2))$$

is a term and

$$\sim ((a := 1)(b := 0) = -(c := 2))$$

$$(a := 1) + (b := 0) \Rightarrow (c := 2)$$

are formulas.

4. Semantics of the decision language. Semantics of the decision language is a function which assigns the meaning to terms and formulas. The meaning of a term is subset of objects from the universe U obeying the properties expressed by the term; the meaning of a formula is the true or false.

The formal definition of semantics is as follows: the meaning of a term in L_S with respect to the decision table $S=(U, C, D, V, f)$ is the function g_S (denoted g – when S is understood), defined inductively with respect to the complexity of the term, as shown below

A1) $g(0)=\emptyset, g(1)=U$

A2) $g(a:=v)=\{x \in U: f_x(a)=v\}$

A3) $g(-t)=U-g(t)$

A4) $g(t.s)=g(t) \cap g(s)$

A5) $g(t+s)=g(t) \cup g(s)$.

The meaning of formulas is the function h_S (or in short h) defined inductively, thus

B1) $h(T)=T, h(F)=F$

B2) $h(t=s)=\begin{cases} T, & \text{if } g(t)=g(s) \\ F, & \text{if } g(t) \neq g(s) \end{cases}$

B3) $h(t \Rightarrow s)=\begin{cases} T, & \text{if } g(t) \subseteq g(s) \\ F, & \text{otherwise} \end{cases}$

B4) $h(\sim \Phi)=\begin{cases} T & \text{if } h(\Phi)=F \\ F, & \text{if } h(\Phi)=T \end{cases}$

B5) $h(\Phi \vee \Psi)=h(\Phi) \vee h(\Psi)$

B6) $h(\Phi \wedge \Psi)=h(\Phi) \wedge h(\Psi)$.

If $h(\Phi)=T$ we say that Φ is **true** in S; if $h(\Phi)=F$ then Φ is said to be **false** in S. If Φ is true (false) in S we write $\vdash_S \Phi$ ($\nvdash_S \Phi$). We omit the subscript S if S is understood.

For example the meaning of the term $(a:=1)(b:=0)$ in the decision table No. 1 is the set $\{1, 4\}$.

The formula $(a:=1)=(b:=2)$ is false and the formula $(a:=1) \Rightarrow (a:=1)$ is true in the decision table No. 1.

As axioms for thus defined language we assume a substitution of the axioms of Boolean algebra for terms and substitutions of the propositional calculus axioms – for formulas. Moreover, the following specific axiom will be assumed for terms

$$(a:=v)=-\sum_{u \neq V,\, u \in V_a} (a:=u).$$

Let $S=(U, C, D, V, f)$ be a decision table, L_S – the decision language and $B \subseteq C \cup D$.

Term $t \in L_S$ is ***B*-elementary** if $t=\prod_{a \in B} (a:=v)$.

A term $t \in L_S$ is in **B-normal form** if $t = \sum s$, where s are some B-elementary terms in L_S.

Property 1. For every term $t \in L_{S/B}$ there exists the term $s \in L_{S/B}$ in B-normal form such that $\vdash t = s$.

Subset $X \subseteq U$ is said to be **B-describable** in L_S if there exists a term $t \in L_{S/B}$ such that $h_S(t) = X$. The term t is called the **B-description** of X in L_S.

Property 2. Subset $X \subseteq U$ is B-describable in L_S iff X is B-definable in S.

Obviously, B-elementary terms in L_S are descriptions of B-elementary sets in S.

5. Decision rules in L_S. Let $S = (U, C, D, V, f)$ be a decision table, $t \in L_{S/C}$ and $s \in L_{S/D}$.

Each formula of the form $t \Rightarrow s$ will be called a decision rule in L_S.

The terms t and s are called the **condition** and the **decision** of the decision rule, respectively.

Two decision rules $r \Rightarrow s$ and $p \Rightarrow q$ in L_S are **equivalent** in S if $h(r) = h(p)$ and $h(s) = h(q)$.

A decision rule $r \Rightarrow s$ in L_S is **deterministic** in S if D-normal form of s is a D-elementary term; otherwise the decision rule is **nondeterministic**.

Property 3. A decision rule $t \Rightarrow s$ in L_S is **true** in S iff all $C \cup D$-elementary terms occurring in $C \cup D$-normal form of t occurs in $C \cup D$-normal form of S.

6. Decision algorithms. A **decision algorithm** in L_S is a finite set of decision rules in L_S.

A decision algorithm is **deterministic** if all its decision rules are deterministic; otherwise the decision algorithm is **nondeterministic**.

With every decision algorithm $\mathfrak{A} = \{t_i \Rightarrow s_i\}$ $0 \leqslant i \leqslant m$ we associate the formula

$$\Psi_{\mathfrak{A}} = \bigwedge_{i=1}^{m} t_i \Rightarrow s_i$$

called the **decision formula** of $\mathfrak{A}$.

The decision algorithm $\mathfrak{A}$ in L_S is said to be **correct** in S if $\Psi_{\mathfrak{A}}$ is true in S; otherwise the decision algorithm $\mathfrak{A}$ is **incorrect** in S.

Property 4.

$$\vdash \bigwedge_{i=1}^{m} (t \Rightarrow s) \quad \text{iff} \quad \vdash \Big(\sum_{i=1}^{m} t_i \Rightarrow s\Big).$$

Two decision algorithms in L_S are **equivalent** in S if both decision algorithms consist of equivalent decision rules in S or the decision rules of both algorithms satisfy the property 4.

INSTITUTE OF COMPUTER SCIENCE, POLISH ACADEMY OF SCIENCES, PKIN, PO BOX 22, 00-901 WARSAW
(INSTYTUT PODSTAW INFORMATYKI PAN)
UNIVERSITY OF NORTH CAROLINA, DEPARTMENT OF COMPUTER SCIENCE, CHARLOTTE, NC

REFERENCES

[1] A. Mrózek, *Information systems and control algorithms*, Bull. Pol. Ac.: Tech., 33 (1985) 195–204.

[2] Z. Pawlak, *Rough sets*, Int. J. Inf. Comp. Sci., 11 (1982) 341–356.

[3] Z. Pawlak, *Rough classification*, Int. J. Man-Machine Studies, 20 (1984) 469–48[illegible].

[4] S. Pollack, H. Hicks, W. Harrison, *Decision tables: theory and practice*, Wiley and Sons Inc., New York 1971.

З. Павляк, **Таблицы принятия решений и решающие алгоритмы**

В работе доказывается, что использование приближенных множеств для анализа таблиц принятия решений ведет к простому методу, по которому можно проверить, является ли таблица детерминированной или нет. Кроме того, доказывается, что такой подход может применяться к упрощению таблиц и решающих алгоритмов.